UNCLASSIFIED

AD 417470

DEFENSE DOCUMENTATION CENTER

FOR

SCIENTIFIC AND TECHNICAL INFORMATION

CAMERON STATION, ALEXANDRIA, VIRGINIA

UNCLASSIFIED

NOTICE: When government or other drawings, specifications or other data are used for any purpose other than in connection with a definitely related government procurement operation, the U. S. Government thereby incurs no responsibility, nor any obligation whatsoever; and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use or sell any patented invention that may in any way be related thereto.

64-2

CATALOGED BY DDC
AS AD No. 417470

417470

MEASURES OF SYNTACTIC COMPLEXITY

BY

Y. Bar-Hillel, A. Kasher and E. Shamir

Applied Logic Branch

The Hebrew University of Jerusalem

Technical Report No. 13

This report was prepared for the

U.S. OFFICE OF NAVAL RESEARCH, INFORMATION SYSTEMS BRANCH

under contract No. 62558-3510, NR 049-130

Jerusalem, Israel

August 1963

MEASURES OF SYNTACTIC COMPLEXITY

BY

Y. Bar-Hillel, A. Kasher and E. Shamir

Applied Logic Branch

The Hebrew University of Jerusalem

Technical Report No. 13

This report was prepared for the

U.S. OFFICE OF NAVAL RESEARCH, INFORMATION SYSTEMS BRANCH

under contract No. 62558-3510, NR 049-130

Jerusalem, Israel

August 1963

M E A S U R E S O F S Y N T A C T I C C O M P L E X I T Y

by

Y. Bar-Hillel, A. Kasher and E. Shamir

Section 0. Introduction

The aim of this report is twofold: 1) to establish certain formal connections between three explicata for the concept of syntactic complexity, that have been discussed in recent literature: degree of nesting, degree of self-embedding, and depth of postponed symbols. 2) to present a formal (and trivial) proof of what has been called elsewhere [2, p. 13] the Anti-Wittgensteinian Thesis: Not everything that can be said at all can be said by using syntactically simple sentences exclusively. This thesis is shown to hold for all standard propositional logics, under the assumption that they contain means of expressing infinitely many different sentences and under all plausible conceptions of semantic equivalence; it is also proved true for a certain calculus of arithmetical functions.

Section 1. Degree of nesting and depth of postponed symbols

We shall follow, with minor modifications, the notations and terminology of [3].

V is a given set - the vocabulary. Elements of V will be called symbols and denoted by capital Latin letters. Finite sequences of symbols of V - including the empty sequence - will be called strings over V and denoted by small Latin letters. Sets of strings over V will be called languages over V and denoted by L with subscripts, in general. If $x \in L$, we say that x is a sentence of L. The set of all strings over V will be denoted by W_V. The length of the string x is the number of its symbol occurrences.

By a grammar we understand a finite system of rules determining a language. Grammars will be denoted by Gothic capitals.

Definition 1.1 (a) A context-free grammar (CFG)* is an ordered quadruple $\mathcal{G} = (V, P, T, S)$, where

(i) V is a finite vocabulary;

(ii) P is a finite set of productions of the form $X \to x$ where $X \in V$, $x \in W_V$, $x \neq X$;

(iii) T is a subset of V (the terminal vocabulary), none of whose elements occur on the left side of a production of P;

(iv) S (the initial symbol) is a distinguished element of $V - T$ (the auxiliary vocabulary).

(b) y directly generates z ($y \Rightarrow z$), iff $y = uXv$, $z = uxv$ and $X \to x \in P$.

(c) y generates z ($y \overset{*}{\Rightarrow} z$), iff there exists a sequence of strings $z_0, z_1, \ldots, z_r$ ($r \geq 0$), such that

$$y = z_0,\ z_r = z \text{ and } z_{i-1} \Rightarrow z_i \quad (i = 1, \ldots, r).$$

The sequence $z_0, \ldots, z_r$ will be called a generation tree of z from y.

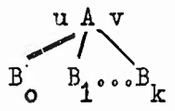
(d) x is a sentence generated by $\mathcal{G}$ iff x is a string over T, and $S \overset{*}{\Rightarrow} x$. $L(\mathcal{G})$ is the set of all sentences generated by $\mathcal{G}$.

(e) A language is representable by a CFG, or is a context-free language (CFL), iff there exists a CFG $\mathcal{G}$, such that $L = L(\mathcal{G})$.

---

*Also called a simple phrase-structure grammar (SPG).

Let us represent direct generation of the form $uAv \Longrightarrow uB_0B_1 \ldots B_k v$ by the following schema:



Then any generation $X \overset{*}{\Longrightarrow} x$ may be represented graphically by a labelled generation-tree (alternatively: tree, phrase marker [4]), cf. Fig 1. Such a tree is essentially a branching pattern, consisting of nodes each of which is labelled by a symbol of V. Referring to the direct generation given above, each node labelled B_j $(1 \leqslant j \leqslant k)$ is said to follow the node labelled A. In any tree there is one node, and only one, which does not follow any other; this node is called the root of the tree. A terminal node is one which has no followers.

For certain purposes it is useful to introduce a special notation for the nodes. Denote the root by 0, and its followers, from left to right (or, conversely, from right to left - we shall have occasion to use this converse notation) by $00, 01, \ldots, 0k$, etc. In general, if a node with $k+1$ followers is denoted by a multi-index α, then its followers, from left to right [from right to left], are denoted by $\alpha 0, \alpha 1, \ldots, \alpha k$.

If we order the multi-indices lexicographically, we thereby induce an ordering of the nodes (cf. Fig. 2). It is clear that in order to obtain the string x whose generation is represented by the tree, we must concatenate the symbols labelleing the terminal nodes, according to the above order.

A path in a tree is a sequence of nodes, beginning at the root, such that each node (except the root) follows its predecessor in the sequence. A path is complete if its last node is terminal (i.e., it leads to a terminal node).

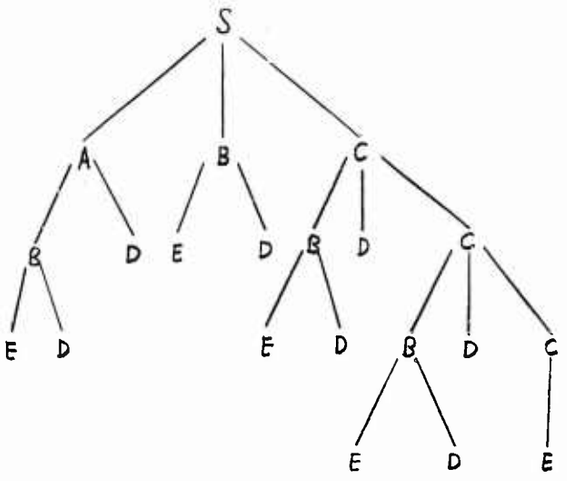


Figure 1

The productions taking part are:

S --> ABC, A --> BD, B --> ED,

C --> BDC, C --> E.

(The string generated is EDDEDEDDEDDE)

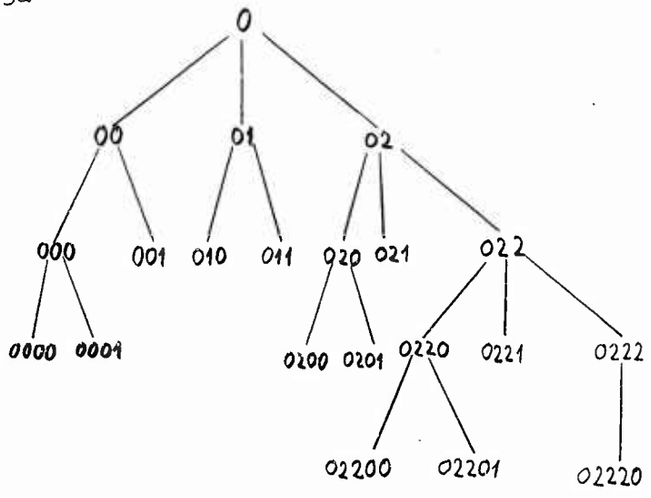
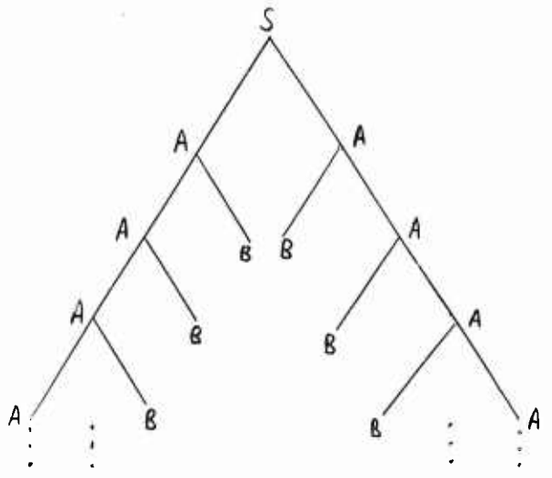
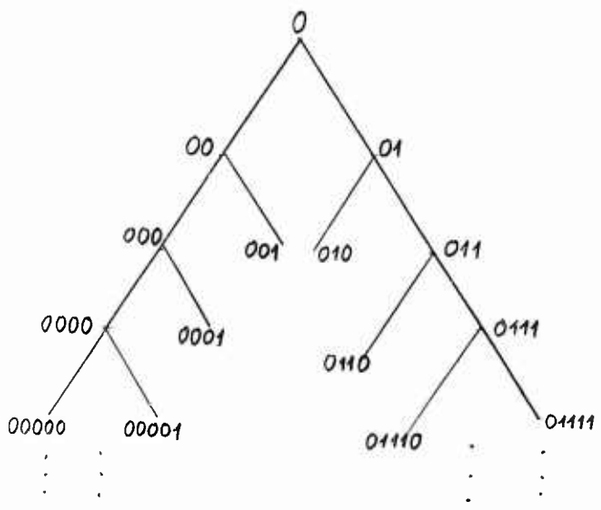


Figure 2

(The same tree as in Figure 1)



(a)



(b)

Figure 3

(b) shows that the depth of left-postponed symbols may be made arbitrarily large. (A similar figure gives the result for ρ)

Clearly, each [terminal] node in a tree uniquely determines a [complete] path - i.e., the path leading to it.

The two extreme (i.e., the leftmost and the rightmost) terminal nodes and all the nodes on the paths leading to them are called the boundary nodes of the tree. All the others are called inner nodes.

Each node N in a tree determines in a natural way a subtree rooted in N. This subtree, when considered in isolation from the nest of the original tree, is itself a tree; the relativization of the above concepts to the subtree is self-evident.

Returning to the multi-index notation, the absolute value of a multi-index α (and of the corresponding node) is defined to be the sum of the indices in α. Omitting from a multi-index α all indices except the last, the absolute value of a node N is then the sum of all the indices of the nodes on the path leading to N. This last notation (i.e., using only indices) was that used by Yngve [14]. Following his lead, as elaborated by Chomsky [7], we define

Definition 1.2. Let Γ be a tree; then the depth of left-postponed symbols of Γ, $\lambda(\Gamma)$, is the maximum absolute value of its nodes. The depth of right-postponed symbols, $\rho(\Gamma)$, is defined in a similar manner, using the converse multi-index notation (i.e., counting the nodes from right to left). The significance of these concepts will be discussed only after the introduction of two other concepts, due to Chomsky [6,7], and after the establishment of some of their basic properties.

Definition 1.3. A node of a tree is nesting [self-embedding (SE)] iff the subtree rooted therein contains a non-terminal [similarly labelled] inner

node (with respect to that same subtree). The two nodes form a nested [self-embedded] pair.

Definition 1.4. (a) The degree of nesting, $\nu(\Gamma)$, of a (terminal) tree Γ is the largest integer m having the following property:

There is a complete path in Γ through m+1 nodes, $N_0, N_1, \ldots, N_m$, where each N_i $(1 \leqslant i \leqslant m)$ is an inner node in the subtree rooted in N_{i-1}.

(b) If the nodes N_i $(1 \leqslant i \leqslant m)$ are all similarly labelled, the integer thus defined is called the degree of self-embedding, $\varepsilon(\Gamma)$.*

Theorem 1.1. The following inequalities hold:

(a) $\varepsilon(\Gamma) \leqslant \nu(\Gamma)$

(b) $\nu(\Gamma) \leqslant \varrho(\Gamma)$, $\nu(\Gamma) \leqslant \lambda(\Gamma)$.

(c) If M is the number of auxiliary symbols in the grammar, then

$$\varepsilon(\Gamma) \geqslant \left[\frac{\nu(\Gamma)}{M}\right]$$

([x] denotes the integral part of x).

Proof. (a) and (b) are obvious.

To prove (c), note that if $\nu(\Gamma) \geqslant kM$, then there are at least kM+1 nodes which satisfy the condition of Def. 1.4. (a), and all

---

*Note that this definition differs slightly from that given by Chomsky [6, p. 11], because of the inequality in (iv) of his definition. This fact may change slightly a number of results.

of them except the last are labelled by auxiliary symbols. Thus, at least k are similarly labelled, so that $\varepsilon(\Gamma) \geq k = \left[\frac{\nu(\Gamma)}{M}\right]$.

Since (c) implies $\nu(\Gamma) \leq M(\ell(\Gamma) + 1)$, it may be said that, for a given grammar G, $\nu(\Gamma)$ and $\varepsilon(\Gamma)$ are, up to a certain multiplicative constant, equivalent measures. This is not true of $\nu(\Gamma)$ (or $\varepsilon(\Gamma)$) and the depths $\rho(\Gamma)$ and $\lambda(\Gamma)$; for we may have $\nu(\Gamma) = 1$, while both $\rho(\Gamma)$ and $\lambda(\Gamma)$ are arbitrarily large (cf. Fig. 3). Clearly, the two depths $\rho(\Gamma)$ and $\lambda(\Gamma)$ are also incomparable.

Nevertheless, the following result (given without proof) shows that a connection of sorts exists between ρ [λ] and ε :

Theorem 1.2. For every CF-grammar there exist two constants, a_ρ and b_ρ [a_λ and b_λ], depending only upon the grammar, such that for every generation tree Γ of the grammar, the following inequality holds:

$$\rho(\Gamma) \geq a_\rho \varepsilon(\Gamma) + b_\rho \qquad [\lambda(\Gamma) \geq a_\lambda \varepsilon(\Gamma) + b_\lambda].$$

(Equality may be realized in certain trees.).

Consider now a tree Γ_1, one of whose terminal nodes, N, is "replaced" by another tree Γ_2 (N being relabelled by the label of the root of Γ_2). Call the resulting composed tree Γ (cf. Fig. 4).

Theorem 1.3. The following inequalities hold:

(a) $\rho(\Gamma) \leq \rho(\Gamma_1) + \rho(\Gamma_2)$; $\lambda(\Gamma) \leq \lambda(\Gamma_1) + \lambda(\Gamma_2)$

(b) $\nu(\Gamma) \leq \nu(\Gamma_1) + \nu(\Gamma_2) + 1$

$\varepsilon(\Gamma) \leq \varepsilon(\Gamma_1) + \varepsilon(\Gamma_2) + 1$

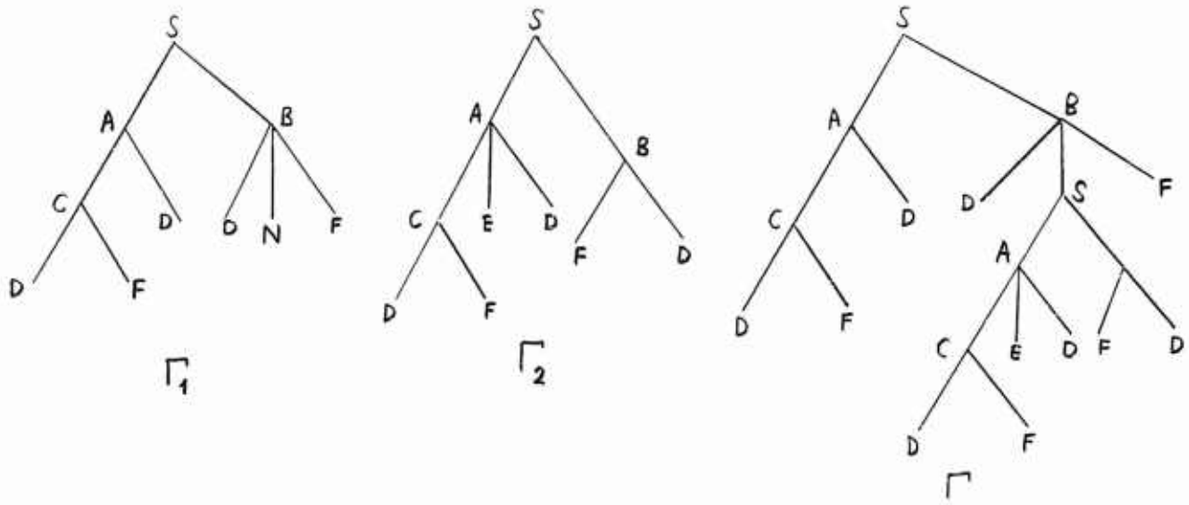


Figure 4

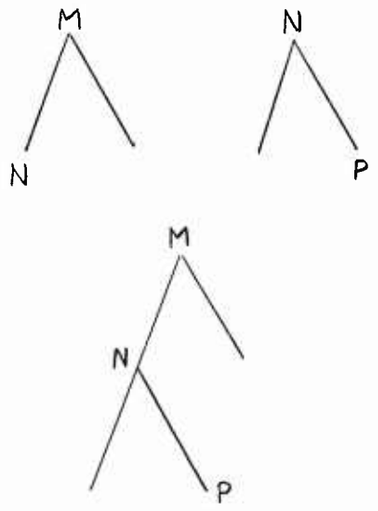


Figure 5

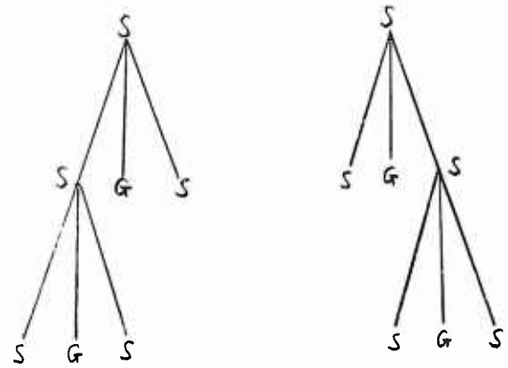


Figure 6

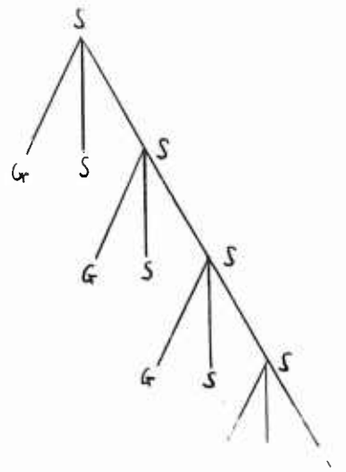


Figure 7

Moreover, if N is on a path which determines the respective depth or degree of Γ_1, then equality holds in (a) and either equality holds in (b) or $\nu(\Gamma) = \nu(\Gamma_1) + \nu(\Gamma_2)$.

Proof. All the assertions are obvious. It is only necessary to explain why $\nu(\Gamma)$ (or $\varepsilon(\Gamma)$) may be larger by one than the sum. This may occur if there are three nodes, M, N and P, on the same path in Γ, such that neither (M,N) nor (N,P) is a nesting pair, but (M,P) is (cf. Fig. 5).

Note that the additivity of the second part of the theorem is in general not true for $\varepsilon(\Gamma)$, since the labels of the longest SE sequences in Γ_1 and Γ_2 may be different.

The concepts ρ, λ, ε and ν introduced above as measures of the complexity of trees emerged in the work of Yngve [14] and Chomsky [6,7], who were engaged in the construction of models to explain the linguistic behavior of users of natural languages. A detailed discussion and comparison of both approaches and an explanation of the way they arise appears in Chomsky [7] (cf. also Bar-Hillel [2, Second Lecture]). Here we shall briefly summarize some basic points.

Both authors assume that the model should be presented as a strictly finite device $\mathfrak{D}$, which is usually called a finite automaton with output (or a finite transducer). This device operates on the sentences of the language (as either inputs or outputs) from left to right. Moreover, it is assumed that the "permanent memory" of $\mathfrak{D}$ contains a CF-grammar of the language.

Now in the Yngve model (which was originally intended as a description of the speaker's behavior) the device $\mathfrak{D}$, in order to generate a sentence,

constructs generation trees from root to terminal nodes and from left to right. This implies that whenever a direct generation $A \Rightarrow B_0 \cdots B_k$ is reached, in which B_j is the leftmost non-terminal, then the terminals $B_0, \cdots, B_{j-1}$ are added to the output, whereas $B_{j+1}, \cdots, B_k$ are transferred to the temporary memory and the branching of B_{j+1} is postponed till the subtree rooted in B_j is fully generated.

Now, it is easily verified that $\rho(\Gamma)$, the depth of right-postponed symbols, is the maximum number of symbols in the temporary memory during the generation of Γ. In other words, if the capacity of the temporary memory is fixed and equal to k, then $\mathcal{D}$ can handle only trees such that $\rho(\Gamma) \leq k$.

If we assume that $\mathcal{D}$ deals with trees in the opposite direction, from terminals to root (but again from left to right), then the role of $\rho(\Gamma)$ will be played by $\lambda(\Gamma)$, the depth of left-postponed symbols.

Note that for $\rho(\Gamma)$ to remain bounded, only a limited number of symbols which are not rightmost may branch; on the other hand, rightmost symbols may branch arbitrarily often. Thus the tree is, of necessity, predominantly right-recursive. Similarly, trees for which $\lambda(\Gamma)$ is bounded are predominantly left-recursive.

Suppose now that no restrictions are imposed upon the manner in which $\mathcal{D}$ operates on trees. Then the degree of nesting emerges as a natural measure. Indeed, the following assertion results from Chomsky's work [5]:

It is possible to construct a device Ψ and a strictly monotonic function $g_1(m)$, such that, if the available computing space of Ψ is m, then Ψ can handle those, and only those, trees Γ (of a grammar G) for which $\nu(\Gamma) \leq g_1(m)$.

In view of the equivalence of the measures $\nu(\Gamma)$ and $\varepsilon(\Gamma)$, the same result (with a different function, $g_2(m)$) holds for $\varepsilon(\Gamma)$; indeed, Chomsky's results are usually formulated for $\varepsilon(\Gamma)$.

A critique of Yngve's more restrictive assumptions, which lead to $\rho(\Gamma)$ or $\lambda(\Gamma)$, is given in Chomsky [7]. But, even for unlabelled trees, there is no reason why right or left-recursion should be singled out, and the degree of nesting seems to us a more adequate measure of complexity.

We add a few remarks explaining why the degree of nesting is in a sense a more natural measure than the degree of self-embedding.

The degree of nesting (like the two depths) depends only on the tree and not on the labelling. This is not so in the case of the degree of SE. For this reason, $\nu(\Gamma)$ compares better with $\rho(\Gamma)$ and $\lambda(\Gamma)$, which are themselves useful in many cases. This accounts also for the lack of additivity for $\varepsilon(\Gamma)$ in Theorem 1.3. This kind of additivity seems, however, to be a natural formal requirement for the adequacy of a proposed measure of complexity.

Another consequence of this dependence on labelling is that $\varepsilon(\Gamma)$ is very unstable, even for slight changes in the grammar. For instance, consider a tree Γ for which $\varepsilon(\Gamma) = k$ and A is the only SE symbol. Let $A = A_1$ and add to the grammar a replica A_2 of A. Assuming, for the sake of convenience, that k is even, replace every second occurrence of A on every path by A_2. Clearly, ε is decreased thereby to $\frac{k}{2}$, while ν is not changed. In a similar manner, ε may be decreased m-fold, for any natural m, by taking m replicas of A.

To summarize, it seems rather clear that nesting is the main phenomenon responsible for syntactic complexity of sentences. It is nesting which

creates a dependence between the parts to the left and to the right of the nested element.*

Section 2. Indispensability of arbitrarily complex sentences in propositional logics

Our aim in this section is to show that, in general, unbounded complexity of sentences is essentially indispensable. To this purpose, we shall prove that certain very elementary propositional logics contain sentences of arbitrarily high degrees of nesting, which are not (semantically) equivalent to any sentences of lower degree. This result also holds, albeit with certain restrictions, for the other measures of syntactic complexity discussed in §1.

One reason for our interest in (various) propositional logics, apart from the fact that their formation rules are easily expressed as CF-grammars, is that some of their sentential connectives have syntactically close analogues in natural languages; this is not true to the same extent for the additional symbols of, say, quantification logic. Thus our results are of rather direct significance for natural languages. Indeed, the simplest way (though not the deepest) of exhibiting complexity and various other phenomena in natural languages has often been by recourse to tricks of propositional logic.

---

*It is natural to assume that this dependence creates a load on memory and concentration. What additional tensions, if any, are created by self-embedding, in addition to those caused by nesting, remains to be investigated.

All propositional calculi compared here will contain some or all of the following generation rules:

(1) $S \rightarrow FS$

(2) $S \rightarrow [SGS]$ or (2') $S \rightarrow GSS$

(3) $S \rightarrow T$

(4) $T \rightarrow T'$

(5) $T \rightarrow P$

In addition, each particular calculus will contain rules requiring that G [and in the case of P1, P3, also F] be rewritten as a binary [unary] connective. In particular, we consider:

P1. (Propositional calculus in Russell notation): (1) - (5), and:
$F \rightarrow \sim$, $G \rightarrow \supset$, $G \rightarrow \vee$, $G \rightarrow \&$ (negation, implication, disjunction and conjunction, respectively).

P2. (Equivalence calculus): (2) - (5), and: $G \rightarrow \equiv$ (equivalence).

The corresponding calculi in Polish notation are:

P3. (1), (2'), (3) - (5) and: $F \rightarrow N$, $G \rightarrow C$, $G \rightarrow A$, $G \rightarrow K$.

P4. (2'), (3) - (5) and: $G \rightarrow E$.

In all these calculi, rules (3) - (5) are used (following Curry) to form an infinite number of propositional variables (pv): P, P', P", etc. It will be convenient to use informally P_o instead of P, P_1 instead of P', etc.

Remark: In the usual formulations of propositional calculi, one assumes the availability of infinitely many propositional letters, say P_o, P_1, P_2, ... In this way, the uninteresting complexity which may arise from the construction of P', P", etc. is avoided. In fact, the rule $T \rightarrow T'$ is left-recursive,

so that its repeated use increases ρ, the depth of right-postponed symbols. However, it is possible to use instead the rule $T \rightarrow {}'T$, which is right-recursive and so increases the other depth, λ. If it is desired to avoid both contingencies, the relevant grammars may be regarded as CF-grammars with an infinity of terminal symbols. The basic properties of CF-grammars remain unchanged thereby.

Note that any generation in the above grammars may be obtained by first using rules (1) - (2) (or (1) - (2')) (in P1, P3) to construct the "skeleton" of the sentence and then using rules (3) - (5) and the substitutions for F and G to differentiate between the various pv and the connectives. It is the trees of the "skeleton grammar" given by rules (1) - (2) (or (1) - (2')) which will essentially determine the complexity. (Cf. the remark for the role of rules (3) - (5) and their effect on the depths.) (Regarding P2 and P4, only rule (2) (or (2')) is used in the skeleton grammar. This is to be understood in the sequel.)

It is easily established that the grammars P1 - P4 are <u>monotectonic</u>, i.e., every sentence has a unique derivation tree. Thus there is no ambiguity in simply speaking of the degree of nesting (or depth) of a sentence x, meaning thereby the degree of nesting (or depth) of its unique derivation tree Γ_x. This would no longer be true if rule (2) were replaced by

(2") $S \rightarrow SGS$

since then $SGSGS$, for example, would have two generation trees (cf. Fig. 6). For the connection between monotectonicity (i.e., syntactic non-ambiguity) and syntactic complexity, see [2, Second Lecture].

Consider a string x generated by rules (1) - (2). Denote by $s(x)$, $g(x)$,

$f(x)$ the number of occurrences in x of the symbols S, G, F, respectively. Similar notations are used for a string y generated by rules (1) - (2'). (In the following, we shall adhere to the convention whereby x is a string generated by (1) - (2), and y - by (1) - (2').) Clearly, $s(x) = g(x) + 1$, and $\nu(\Gamma_x) = \xi(\Gamma_x)$ - the degrees of nesting and of self-embedding coincide, since there is only one branching symbol, S.

Lemma 2.1. *If Γ_x is the generation tree of x, then:*

(a) $\nu(\Gamma_x) = k \Rightarrow k \leq g(x) \leq 2^k - 1$

(b) $\nu(\Gamma_y) = k \Rightarrow k \leq g(y)$

and every value of g permitted by these inequalities may be realized effectively.

Proof. The proof of (a) is by induction on k.

The case $k = 1$ is trivial, since the only strings x for which $\nu(\Gamma_x) = 1$ are $[SGS]$, and any other string obtained from this string by use of rule (1) alone, which introduces no G's.

Assume the theorem true for a given k, and let $\nu(\Gamma_x) = k$. Then ν may be increased to $k+1$ in a variety of ways; the two extreme cases are the following: (1) Rule (2) is applied to a single S on a path determining the degree - if x' is the resulting string then $g(x') = g(x) + 1$. (2) Rule (2) is applied to each of the $g(x) + 1$ occurrences of S in x; each application introduces a new G, and thus $g(x') = 2g(x) + 1 \leq 2(2^k - 1) + 1 = 2^{k+1} - 1$ (by the induction hypothesis). Obviously, all intermediate values may be obtained by appropriate variations.

As for (b), note that an increase in $g(y)$ may or may not entail an

increase in ν . Indeed, in $S \to \underline{G}SS \Rightarrow GGSSS$ the degree of nesting increases at the second stage, whereas in $S \to \underline{G}SS \Rightarrow GSGSS$ there is no increase. (The branching symbol is underlined in each case.) Thus (b) is proved, by a trivial induction. Note that no upper bound (in terms of ν) is possible for $g(y)$ - cf. the tree in Fig. 7, where $\nu = 1$ and g may assume any desired value.

Thus, for rules (1) - (2), $\nu(\Gamma_x)$ is not more than $g(x)$ and not less than $\log_2(g(x) + 1)$. No similar lower bound exists for $\nu(\Gamma_y)$ (rules (1) - (2')), while it may assume any value $\leq g(y)$.

Similar (though slightly more complicated) bounds may be obtained for $\rho(\Gamma_x)$. For $\rho(\Gamma_y)$, again, only an upper bound is obtained (cf. Fig. 7, where $\rho = 2$ and $g(y)$ is not bounded). In the computations for λ it is necessary to take $f(x)$ into account, since the value of λ is increased by the application of the rule $S \to FS$.

Passing now from the skeleton grammars to any one of P1 - P4, $s(x)$ is the number of pv in the generated sentence x , $f(x)$ the number of unary and $y(x)$ the number of binary connectives. By Theorem 1.3 (a), the depth is not increased by application of the rule $T \to T'$, and ρ - by application of $T \to 'T$ (cf. the remark on page 11).

We shall now turn to the semantics of the propositional logics. First, consider the sentences of P1 as truth functions; i.e., assign to each pv the "truth-value" 1 or 0 (<u>true</u> or <u>false</u>), and define the functions $\sim p_1$, $p_1 \supset p_2$, $p_1 \,\&\, p_2$, $p_1 \vee p_2$, $p_1 \equiv p_2$, whose values are again 1 or 0, in the usual manner (For instance, $val(p_1 \supset p_2) = 0$ if $val(p_1) = 1$ and $val(p_2) = 0$; otherwise $val(p_1 \supset p_2) = 1$.) Two sentences are termed

equivalent iff they correspond to identical truth-functions.

Denote by q_n the sentence $[p_0 \supset [p_1 \supset \ldots \supset [p_{n-1} \supset p_n] \ldots]]$.

Theorem 2.1. q_n is not equivalent to any sentence of P1 which contains less than n+1 occurrences of pv. The same is true of any sentence x obtained from q_n by rebracketing.

Proof. It is easy to see that $val(q_n) = 0$ iff $val(p_0) = val(p_1) = \ldots = val(p_{n-1}) = 1$ and $val(p_n) = 0$. A change in the value of any one variable changes the value of the sentence and renders it true. If y is a sentence containing less than n+1 occurrences of pv, then one of the variables of q_n does not occur therein, and a change in the value of this variable alone cannot change val(y). Hence q_n is not equivalent to y.

The same argument proves the assertion not only for q_n, but also for any sentence x containing n+1 pv and having the following property: For any variable P in x there exists an assignment of truth-values to the remaining n variables such that a change in val(p) will change val(x) (the remaining n truth-values remaining fixed).

It will now be proved, by induction on n, that any sentence x obtained from q_n by rebracketing has this property. The assertion is trivial for $n = 2$. In general, x has the form $[y \supset z]$. If the chosen variable P occurs in z, choose an assignment of truth-values in y which renders y true. Then $val([y \supset z]) = val(z)$. By induction, z has the required property, and so has x. If P occurs in y, choose an assignment which renders z false, and then $val([y \supset z]) = 1 - val(y)$. Again by induction, y has the required

property, and so has x.

The Theorem is thus proved.

Remark. A calculus is called m-valued ($m \geq 2$) when its truth-functions are allowed to take any integer truth-value t, where $1 \leq t \leq m$.

For the negation-implication m-valued calculus, truth-values of formulas are determined as follows [12]: (i) $val(\sim P) = m+1-val(P)$

(ii) $val(P \supset Q) = \max(1, 1+val(Q) - val(P))$.

The results of this chapter hold, with some slight modifications, for these calculi as well.

It is well known that two sentences x and y (in P1) are equivalent iff $[[x \supset y] \ \& \ [y \supset x]]$ is derivable from an appropriate set of axioms of the propositional calculus. Similarly, P3 (Polish notation), x and y are equivalent iff $KCxyCyx$ is derivable from an appropriate set of axioms.

Using now Lemma 2.1 and Theorem 2.1:

Theorem 2.2. For every natural number n it is possible to construct a sentence x in P1 or P3 such that $\nu(\ulcorner x \urcorner) = n$ and such that no sentence y for which $\nu(\ulcorner y \urcorner) < n$ is equivalent to x. The same assertion holds for λ and ζ.

Proof. For P1, take $x = q_n$. Obviously, $\nu(\ulcorner q_n \urcorner) = n$. By Theorem 2.1, x is not equivalent to any sentence containing less than $n+1$ occurrences of pv and, as may be demonstrated by methods of the propositional calculus, it is not equivalent to any sentence containing more than $n+1$ pv. Moreover, by similar methods it is easy to see that x is not equivalent to any sentence obtained from itself by rebracketing (which may have a smaller degree of nesting).

The assertion is thus proved for P1.

For P3 take $x = r_n^*$, where:

$$r_n^* = \underbrace{C \dots C}_{n} P_0 P_1 \dots P_n$$

(corresponding to $r_n = [\dots[[p_0 \supset p_1] \supset p_2] \dots \supset p_n]$ in P1 notation).

The sentences q_n and r_n, or their counterparts q_n^* and r_n^* in Polish notation, give the required result for λ and ς, respectively.

The same results regarding the degree of nesting ν may be proved using other definitions of equivalent sentences. Indeed, the same proof, based on Theorem 2.1, gives the result for ν for the equivalence calculus, whose axioms were given by Leśniewski [11, p. 16] (the grammar is now P2), and for the same in Polish notation (grammar P4).

Other possibilities are:

a) x is equivalent to y iff $\vdash_H [x \equiv y]$, where H is the intuitionistic calculus of Heyting [9].

b) x is equivalent to y iff $\vdash_E [x = y]$, where E is the entailment calculus of Anderson-Belnap (cf., e.g., [1]).

c) x is equivalent to y iff $(En)[\vdash_{P_n} [x \equiv y]]$, where P_n is the n-valued logic of Łukasiewicz (cf., e.g., [12]).

d) x is equivalent to y iff $\vdash_S \Box[x \equiv y]$, where S is any modal logic of Lewis or Fitch [10; 8].

In all these cases (except c) the result follows either from the fact that the relevant concept of equivalence is more restricted than in P1 with its usual axioms, or by direct recourse to the valuation result in Theorem 2.1. (The extension of this theorem to multivalued logic proves the result for c.)

Note, however, the following result:

Theorem 2.3. Every sentence x in the equivalence calculus P2 is equivalent to a sentence y for which $\lambda(\Gamma_y) = 1$. The same is true for ρ, and for Polish notation (P4).

This result is easily derived from a theorem of Łukasiewicz:
$[p \equiv [q \equiv r]] \equiv [[p \equiv q] \equiv r]$ (cf. [13, p. 4, footnote 2; 11, p. 16]).

Remarks. (1) It is well known that in Russell's notation any sentence is equivalent to a sentence containing only the connectives "$\sim$" and "&", or "$\sim$" and "$\vee$". However, as is easily seen, this transformation does not reduce the degree of nesting.

(2) The sentence q_n is equivalent to
$[|P_0 \,\&\, [P_1 \,\&\ldots\&\, [P_{n-2} \,\&\, P_{n-1}]\ldots]] \supset P_n]$ (also according to definitions b and d). There is no harm in omitting brackets and writing
$[[P_0 \,\&\, P_1 \,\&\ldots\&\, P_{n-1}] \supset P_n]$, since the various trees yield equivalent sentences. The complexity of the sentence has no doubt been reduced; however, other sentences, obtained from q_n by rebracketing (e.g., the sentence r_n, $[\ldots[[P_0 \supset P_1] \supset P_2] \supset \ldots \supset P_n]$) are not so amenable to simplification.

Section 3. Complexity of arithmetical functions

In this section, measures of complexity analogous to those developed in the preceding sections for sentences will be developed for certain functions over the natural numbers. An anti-Wittgensteinian thesis will be proved for a certain representation of these functions. It must be emphasized that the concepts to be introduced are measures of complexity of functions in a certain, well-defined, sense only.

Define a CF-grammar as follows:

The vocabulary consists of

(i) the initial symbol T;

(ii) a finite set of auxiliary symbols $x_{i,j}$, where $i=1,\ldots,k$ and, for every i, $j = 1,\ldots,n_i$;

(iii) the terminal symbols $f_1,\ldots,f_k$, y, $z_{i,j}$ (where i and j vary as in (ii)), F, (and) .

N.B. The indices i and j above are not meant to be part of the relevant symbols; they are only a convenient shorthand for differentiating the various x's and z's.

The productions of the grammar are the following:

(1) (i,j) $T \rightarrow x_{i,j}$ (i,j as in (ii) above);

(2) (i,j,m) $x_{i,j} \rightarrow f_m(x_{m,1},\ldots,x_{m,n_m})$ (i,j as above, $m = 1,\ldots,k$);

(3) (i,j) $x_{i,j} \rightarrow F(y,T)$;

(4) $'i,j)$ $x_{i,j} \rightarrow z_{i,j}$.

The trees generated by the above grammar represent "function-terms" in the following way:

$f_1,\ldots,f_m$ are the initial functions, which are fixed for the grammar in question.

Rule (2) is a substitution scheme.

Rule (3) is a definition-by-primitive-recursion scheme; its application is meant to represent the generation of a function defined by

$$F(0) = y,$$

$$F(n') = T(n,F(n)),$$

where T is a term of the grammar containing only two variables from the set of x's, which has already been generated by a tree.

The z's represent numerical variables.

In view of the above naive description, the following rigorous definition is natural:

Definition 3.1. A function f (in the naive sense) is representable in the above grammar, with respect to a system Π of semantic rules, iff there exists a term generated by the grammar whose interpretation by Π is identical (extensionally) with f.

Note that the representation of a function in the above sense is not unique, since it depends upon the generation tree considered.

Theorem 3.1. There exist sets $f_1,\ldots,f_m$ such that for a suitable system Π all the primitive recursive functions are representable in the grammar.

Proof. The assertion is a direct consequence of a result of R.M. Robinson [18], which shows that in order to obtain the primitive recursive functions in the above manner it is sufficient to take $k = 3$, $n_1 = n_2 = 1$, $n_3 = 2$. (Cf. Robinson's paper for the various possible choices of initial functions.)

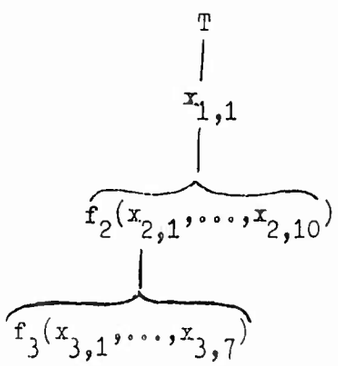
Remark. A similar result, due to J. Robinson [17], hints at the possibility of obtaining all the (general) recursive functions, if the initial functions are supplemented by certain transformations, in a certain sense (such as minimization, inversion, etc.); i.e., it may be possible to obtain representations of all the general recursive functions by means of a transformation grammar, the representations of the primitive recursive functions serving as kernel.

A generation rule P is applied on a branch B of a tree Γ iff there is a node on B such that its direct successors in Γ are obtained by P from that node. (Strictly speaking, this definition should be phrased in terms of the symbols labelling the nodes.)

Definition 3.2. The degree of substitution, R(f), of a function-term f, given as a generation tree, is the maximum number of applications of rules of type (2) (i,j,m) on a branch of the tree.

Definition 3.3. The degree of recursivity, I(f), of a function-term is defined in a way similar to R(f), with regard to applications of rules of type (3) (i,j).

Example. The degree of substitution of the following tree is 2:



The term generated by the above tree is

$$f_2(f_3(x_{3,1},\ldots,x_{3,7}),x_{2,2},\ldots,x_{2,10}).$$

It is evident that these degrees are measures of the complexity, in certain senses, of functions, in a suitable interpretation. The degree of substitution, for instance, is a measure of the need to employ substitutions in order to define a function in the above manner.

A further, more specific, explication of the concept is given by the following definition:

Definition 3.4. Let f be a function (in the naive sense) which is representable with respect to the semantic rules Π. If Γ_f is the set of terms representing f, then the degree of substitution of f is

$$R(f) = \min_{\Gamma \in \Gamma_f} R(\Gamma).$$

The degree of recursivity of a function is defined in a similar way.

Remark. Different explications of the above concepts are obtained if in Definitions 3.2 and 3.3 the degree is defined as the maximum number of applications of the same rule (of the appropriate type) on a branch of the tree. The difference between the two explications is analogous to the difference between degree of nesting and degree of self-embedding (cf. section 1), and will be clarified in the sequel.

Theorem 3.2. If the degree of nesting of the tree Γ with respect to rules of type (2) (i,j,m) is denoted by $\nu^{(2)}(\Gamma)$, and the degree of nesting with respect to rules of type (3) (i,j) by $\nu^{(3)}(\Gamma)$, then $R(\Gamma) = \nu^{(2)}(\Gamma)$

and $I(\Gamma) = \gamma^{(3)}(\Gamma)$. Moreover, if $\mathcal{E}(\Gamma)$ denotes the degree of self-embedding with respect to the symbol T only, then $I(\Gamma) = \mathcal{E}(\Gamma)$.

Proof. Obvious.

Remark. For the explications mentioned in the Remark following Definition 3.4, the first assertion of the above theorem is true for the degree of self-embedding of the tree. In this case, it is clear why the degree of nesting is to be preferred to the degree of self-embedding as an explication of complexity.

The reduction of the degrees defined in Definition 3.4 to the concepts defined in Section 1 will be fully utilized in the proof of Theorem 3.4 below.

Theorem 3.3. A set of tapes generated (or: accepted) by a finite automaton is a primitive recursive set (via an appropriate Gödel-numbering).

Proof. The proof is obtained by means of slight alterations in the proof of the main theorem of Myhill [15, Chapter 4], according to which every set of tapes accepted by a linear bounded automaton is primitive recursive.

Since every finite automaton is a fortiori a linear bounded automaton, the present theorem may be proved using a theorem of Ritchie [16, p. 76] according to which the set of functions computed by a linear bounded automaton is the smallest set which contains the constant functions, the successor and multiplication functions, and is closed under explicit transformations, composition and limited recursion.

Theorem 3.4. Let Π be a system of semantic rules with respect to which all the primitive recursive functions are representable in the grammar. Then, for every n and m, there exists a primitive recursive function $f_{m,n}$ such

that $R(f_{m,n}) > m$ and $I(f_{m,n}) > n$.

I.e., there exists a tree $\Gamma_{m,n}$ such that $R(\Gamma_{m,n}) > m$, $I(\Gamma_{m,n}) > n$ and for every tree Γ which represents the same function (extensionally), $R(\Gamma) \geq R(\Gamma_{m,n}) > m$ and $I(\Gamma) \geq I(\Gamma_{m,n}) > n$.

Proof. It is clear, on the basis of a theorem of Chomsky [6,7], that the set of trees Γ of the grammar, for which $R(\Gamma) \leq m$ or $I(\Gamma) \leq n$, may be generated by a finite automaton; thus, by Theorem 3.3, the enumerating function of the set of these trees, which is defined by the automaton, is primitive recursive, and thus, the diagonal function f' defined with the aid of the enumerating function is primitive recursive. Define $g(n) = f(n)+1$. Then $g(n)$ is also primitive recursive and is thus, by hypothesis, representable with respect to Π.

For any tree Γ representing this function, it is clear that $R(\Gamma) > m$ and $I(\Gamma) > n$. Denote by E the set of these trees. Let E_R be the set of trees Γ' in E such that $R(\Gamma') = \min_{\Gamma \in E} R(\Gamma)$; let E_{IR} be the set of trees Γ' in E_R such that $I(\Gamma') = \min_{\Gamma \in E_R} I(\Gamma)$. It is obvious that any member of E_{IR} may serve as the tree $\Gamma_{m,n}$ whose existence is postulated by the theorem.

REFERENCES

[1] Anderson, A.R., Completeness theorems for the systems E of entailment and EQ of entailment with quantification, Z. f. math. Logik und Grund. d. Math., vol. 6 (1960), pp. 201-216.

[2] Bar-Hillel, Y. Four lectures on algebraic linguistics and machine translation, Hebrew University, Jerusalem, 1963.

[3] Bar-Hillel, Y., Perles M., Shamir, E. On formal properties of simple phrase structure grammars, Z. f. Phonetik, Sprachwissenschaft und Kommunikationsforschung, vol. 14 (1961), pp. 143-172.

[4] Chomsky, N. Three models for the description of language, IRE transactions on information theory, vol. IT-2, Proceedings of the symposium on information theory (1956), pp. 113-124.

[5] Chomsky, N. On certain formal properties of grammars, Information and control, vol. 2 (1959), pp. 137-167.

[6] Chomsky, N. On the notion "rule of grammar", Proceedings of symposia in applied mathematics, vol. 12, Amer. Math. Soc., Providence, R.I., 1961, pp. 6-24.

[7] Chomsky, N. Formal properties of grammars, Handbook of mathematical psychology (D. Luce, E. Bush, F. Galanter, eds.) (in press).

[8] Fitch, F.B. Symbolic logic, New York 1952.

[9] Heyting, A. Intuitionism, Amsterdam 1956.

[10] Lewis, C.I., Langford, C.H. Symbolic logic, New York 1932.

[11] Leśniewski, St. Grundzüge eines neuen Systems der Grundlagen der Mathematik, Fund. Math., vol. 14 (1929), pp. 1-81.

[12] Rosser, J.B., Turquette, A.R. Axiom schemes for m-valued propositional calculi, J. of Symbolic Logic, vol. 10 (1945), pp. 61-82.

[13] Tarski, A. Logic, semantics, metamathematics, Oxford 1956.

[14] Yngve, V.H. A model and an hypothesis for language structure, Proceedings of the American Philosophical Society, vol. 101 (1960), pp. 444-466.

[15] Myhill, J. Linear bounded automata, Wright Air Development Division, Technical Note 60-165, 1960.

[16] Ritchie, R.W. Classes of recursive functions of predictable complexity, Princeton University (mimeographed) 1961.

[17] Robinson, J. General recursive functions, Proc. Amer. Math. Soc., vol. 1 (1950), pp. 703-718.

[18] Robinson, R.M. Primitive recursive functions, Bull. Amer. Math. Soc., vol. 53, (1947), pp. 925-942.